TASK 1 : Database Design

1. Write an SQL query to retrieve the names and emails of all customers.

```
mysql> create database TechShop;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| college            |
| hexprac            |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| school             |
| sql_hr             |
| sql_inventory      |
| sql_invoicing      |
| sql_store          |
| sys                |
| techshop           |
| world              |
+--------------------+
14 rows in set (0.00 sec)
```

(TechShop Database)

2. Define the schema for the Customers, Products, Orders, OrderDetails and Inventory tables based on the provided schema.

   - Customers:

```
mysql> CREATE TABLE Customers (
    ->     CustomerID INT PRIMARY KEY,
    ->     FirstName VARCHAR(255),
    ->     LastName VARCHAR(255),
    ->     Email VARCHAR(255),
    ->     Phone VARCHAR(15),
    ->     Address VARCHAR(255));
Query OK, 0 rows affected (0.03 sec)

mysql> desc Customers;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| CustomerID | int          | NO   | PRI | NULL    |       |
| FirstName  | varchar(255) | YES  |     | NULL    |       |
| LastName   | varchar(255) | YES  |     | NULL    |       |
| Email      | varchar(255) | YES  |     | NULL    |       |
| Phone      | varchar(15)  | YES  |     | NULL    |       |
| Address    | varchar(255) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

(Customer Table)

- Products:

```
mysql> CREATE TABLE Products (
    ->     ProductID INT PRIMARY KEY,
    ->     ProductName VARCHAR(255),
    ->     Description TEXT,
    ->     Price DECIMAL(10, 2));
Query OK, 0 rows affected (0.01 sec)

mysql> desc Products;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| ProductID   | int           | NO   | PRI | NULL    |       |
| ProductName | varchar(255)  | YES  |     | NULL    |       |
| Description | text          | YES  |     | NULL    |       |
| Price       | decimal(10,2) | YES  |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

(Products Table)

- Orders:

```
mysql> CREATE TABLE Orders (
    ->     OrderID INT PRIMARY KEY,
    ->     CustomerID INT,
    ->     OrderDate DATE,
    ->     TotalAmount DECIMAL(10, 2),
    ->     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID));
Query OK, 0 rows affected (0.03 sec)

mysql> desc Orders;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| OrderID     | int           | NO   | PRI | NULL    |       |
| CustomerID  | int           | YES  | MUL | NULL    |       |
| OrderDate   | date          | YES  |     | NULL    |       |
| TotalAmount | decimal(10,2) | YES  |     | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

(Order Table)

- OrdersDetails:

```
mysql> CREATE TABLE OrderDetails (
    ->     OrderDetailID INT PRIMARY KEY,
    ->     OrderID INT,
    ->     ProductID INT,
    ->     Quantity INT,
    ->     FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    ->     FOREIGN KEY (ProductID) REFERENCES Products(ProductID));
Query OK, 0 rows affected (0.03 sec)

mysql> desc OrderDetails;
+---------------+------+------+-----+---------+-------+
| Field         | Type | Null | Key | Default | Extra |
+---------------+------+------+-----+---------+-------+
| OrderDetailID | int  | NO   | PRI | NULL    |       |
| OrderID       | int  | YES  | MUL | NULL    |       |
| ProductID     | int  | YES  | MUL | NULL    |       |
| Quantity      | int  | YES  |     | NULL    |       |
+---------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

(OrderDetails Table)

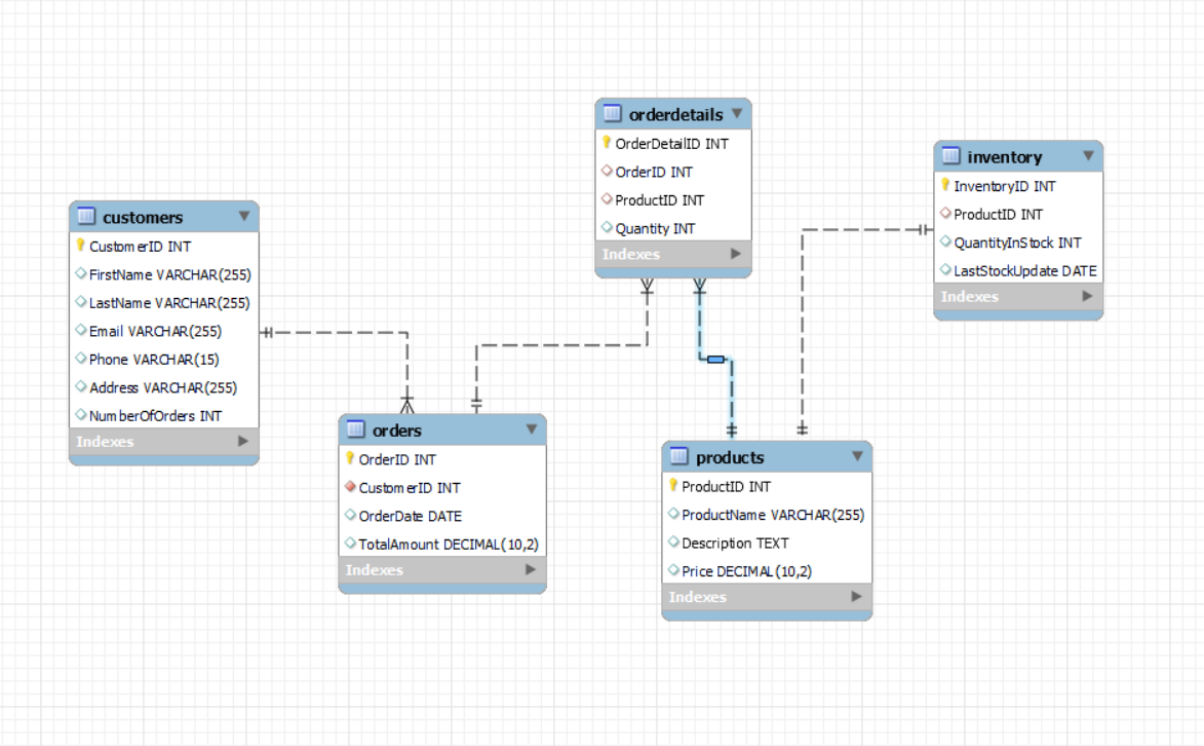
- Inventory:

```
mysql> CREATE TABLE Inventory (
    ->     InventoryID INT PRIMARY KEY,
    ->     ProductID INT,
    ->     QuantityInStock INT,
    ->     LastStockUpdate DATE,
    ->     FOREIGN KEY (ProductID) REFERENCES Products(ProductID));
Query OK, 0 rows affected (0.02 sec)

mysql> desc Inventory;
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| InventoryID     | int  | NO   | PRI | NULL    |       |
| ProductID       | int  | YES  | MUL | NULL    |       |
| QuantityInStock | int  | YES  |     | NULL    |       |
| LastStockUpdate | date | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

(Inventory Table)

3. Create an ERD (Entity Relationship Diagram) for the database.



(Entity Relationship Diagram)

4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

Field	Type	Null	Key	Default	Extra
CustomerID	int	NO	PRI	NULL	

(Customer Table)

Field	Type	Null	Key	Default	Extra
ProductID	int	NO	PRI	NULL	

(Products Table)

Field	Type	Null	Key	Default	Extra
OrderID	int	NO	PRI	NULL	
CustomerID	int	YES	MUL	NULL	

(Order Table)

Field	Type	Null	Key	Default	Extra
OrderDetailID	int	NO	PRI	NULL	
OrderID	int	YES	MUL	NULL	
ProductID	int	YES	MUL	NULL	

(OrderDetails Table)

Field	Type	Null	Key	Default	Extra
InventoryID	int	NO	PRI	NULL	
ProductID	int	YES	MUL	NULL	

(Inventory Table)

5. Insert at least 10 sample records into each of the following tables.

- Customers –

```
mysql> INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone, Address)
    -> VALUES (11, 'Aarav', 'Patel', 'aarav.patel@example.com', '9876543210', '456 Tulsi Lane'),
    -> (12, 'Aditi', 'Sharma', 'aditi.sharma@example.com', '7778889999', '789 Ganges Street'),
    -> (13, 'Arjun', 'Verma', 'arjun.verma@example.com', '5554443333', '234 Himalaya Road'),
    -> (14, 'Avni', 'Gupta', 'avni.gupta@example.com', '2221110000', '567 Taj Avenue'),
    -> (15, 'Rahul', 'Singh', 'rahul.singh@example.com', '6667778888', '890 Mango Lane'),
    -> (16, 'Neha', 'Yadav', 'neha.yadav@example.com', '1112223333', '123 Rose Garden'),
    -> (17, 'Vedant', 'Kumar', 'vedant.kumar@example.com', '9998887777', '678 Lotus Street'),
    -> (18, 'Isha', 'Shah', 'isha.shah@example.com', '3332221111', '345 Peacock Lane'),
    -> (19, 'Kabir', 'Joshi', 'kabir.joshi@example.com', '8889990000', '456 Mango Avenue'),
    -> (20, 'Ananya', 'Das', 'ananya.das@example.com', '4445556666', '789 Jasmine Road');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Customers;
+------------+-----------+----------+------------------------+------------+------------------+
| CustomerID | FirstName | LastName | Email                  | Phone      | Address          |
+------------+-----------+----------+------------------------+------------+------------------+
|         11 | Aarav     | Patel    | aarav.patel@example.com | 9876543210 | 456 Tulsi Lane   |
|         12 | Aditi     | Sharma   | aditi.sharma@example.com | 7778889999 | 789 Ganges Street |
|         13 | Arjun     | Verma    | arjun.verma@example.com | 5554443333 | 234 Himalaya Road |
|         14 | Avni      | Gupta    | avni.gupta@example.com | 2221110000 | 567 Taj Avenue   |
|         15 | Rahul     | Singh    | rahul.singh@example.com | 6667778888 | 890 Mango Lane   |
|         16 | Neha      | Yadav    | neha.yadav@example.com | 1112223333 | 123 Rose Garden  |
|         17 | Vedant    | Kumar    | vedant.kumar@example.com | 9998887777 | 678 Lotus Street |
|         18 | Isha      | Shah     | isha.shah@example.com  | 3332221111 | 345 Peacock Lane |
|         19 | Kabir     | Joshi    | kabir.joshi@example.com | 8889990000 | 456 Mango Avenue |
|         20 | Ananya    | Das      | ananya.das@example.com | 4445556666 | 789 Jasmine Road |
+------------+-----------+----------+------------------------+------------+------------------+
10 rows in set (0.00 sec)
```

(Customer Table)

- Products –

```
mysql> INSERT INTO Products (ProductID, ProductName, Description, Price)
    -> VALUES (11, 'LED TV', '55-inch 4K Smart LED TV', 799.99),
    -> (12, 'Air Conditioner', 'Split-type AC with remote control', 499.99),
    -> (13, 'Refrigerator', 'Double-door frost-free refrigerator', 899.99),
    -> (14, 'Washing Machine', 'Front-load washing machine', 649.99),
    -> (15, 'Microwave Oven', 'Convection microwave with grill', 129.99),
    -> (16, 'Coffee Maker', 'Automatic drip coffee maker', 59.99),
    -> (17, 'Blender', 'High-speed blender with multiple settings', 79.99),
    -> (18, 'Vacuum Cleaner', 'Cordless handheld vacuum cleaner', 149.99),
    -> (19, 'Iron', 'Steam iron with adjustable temperature', 29.99),
    -> (20, 'Hair Dryer', 'Professional hair dryer with ion technology', 39.99);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Products;
+-----------+----------------+-------------------------------------------+--------+
| ProductID | ProductName    | Description                               | Price  |
+-----------+----------------+-------------------------------------------+--------+
|        11 | LED TV         | 55-inch 4K Smart LED TV                   | 799.99 |
|        12 | Air Conditioner | Split-type AC with remote control        | 499.99 |
|        13 | Refrigerator   | Double-door frost-free refrigerator       | 899.99 |
|        14 | Washing Machine | Front-load washing machine               | 649.99 |
|        15 | Microwave Oven | Convection microwave with grill           | 129.99 |
|        16 | Coffee Maker   | Automatic drip coffee maker               |  59.99 |
|        17 | Blender        | High-speed blender with multiple settings |  79.99 |
|        18 | Vacuum Cleaner | Cordless handheld vacuum cleaner          | 149.99 |
|        19 | Iron           | Steam iron with adjustable temperature    |  29.99 |
|        20 | Hair Dryer     | Professional hair dryer with ion technology |  39.99 |
+-----------+----------------+-------------------------------------------+--------+
10 rows in set (0.00 sec)
```

(Products Table)

- Orders –

```
mysql> INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
    -> VALUES (111, 11, '2024-02-01', 1200.00),
    -> (112, 12, '2024-02-02', 850.00),
    -> (113, 13, '2024-02-03', 650.00),
    -> (114, 14, '2024-02-04', 1100.00),
    -> (115, 15, '2024-02-05', 450.00),
    -> (116, 16, '2024-02-06', 1050.00),
    -> (117, 17, '2024-02-07', 400.00),
    -> (118, 18, '2024-02-08', 750.00),
    -> (119, 19, '2024-02-09', 600.00),
    -> (120, 20, '2024-02-10', 950.00);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Orders;
+---------+------------+------------+-------------+
| OrderID | CustomerID | OrderDate  | TotalAmount |
+---------+------------+------------+-------------+
|     111 |         11 | 2024-02-01 |     1200.00 |
|     112 |         12 | 2024-02-02 |      850.00 |
|     113 |         13 | 2024-02-03 |      650.00 |
|     114 |         14 | 2024-02-04 |     1100.00 |
|     115 |         15 | 2024-02-05 |      450.00 |
|     116 |         16 | 2024-02-06 |     1050.00 |
|     117 |         17 | 2024-02-07 |      400.00 |
|     118 |         18 | 2024-02-08 |      750.00 |
|     119 |         19 | 2024-02-09 |      600.00 |
|     120 |         20 | 2024-02-10 |      950.00 |
+---------+------------+------------+-------------+
10 rows in set (0.00 sec)
```

(Orders Table)

- OrderDetails –

```
mysql> INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity)
    -> VALUES (11, 111, 11, 1),
    -> (12, 111, 12, 2),
    -> (13, 112, 13, 1),
    -> (14, 113, 14, 3),
    -> (15, 113, 15, 1),
    -> (16, 114, 16, 2),
    -> (17, 115, 17, 1),
    -> (18, 116, 18, 2),
    -> (19, 117, 19, 1),
    -> (20, 118, 20, 1);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM OrderDetails;
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|            11 |     111 |        11 |        1 |
|            12 |     111 |        12 |        2 |
|            13 |     112 |        13 |        1 |
|            14 |     113 |        14 |        3 |
|            15 |     113 |        15 |        1 |
|            16 |     114 |        16 |        2 |
|            17 |     115 |        17 |        1 |
|            18 |     116 |        18 |        2 |
|            19 |     117 |        19 |        1 |
|            20 |     118 |        20 |        1 |
+---------------+---------+-----------+----------+
10 rows in set (0.00 sec)
```

(OrderDetails Table)

- Inventory –

```
mysql> INSERT INTO Inventory (InventoryID, ProductID, QuantityInStock, LastStockUpdate)
    -> VALUES (11, 11, 8, '2024-02-01'),
    -> (12, 12, 15, '2024-02-02'),
    -> (13, 13, 10, '2024-02-03'),
    -> (14, 14, 5, '2024-02-04'),
    -> (15, 15, 12, '2024-02-05'),
    -> (16, 16, 7, '2024-02-06'),
    -> (17, 17, 10, '2024-02-07'),
    -> (18, 18, 4, '2024-02-08'),
    -> (19, 19, 18, '2024-02-09'),
    -> (20, 20, 6, '2024-02-10');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Inventory;
+-------------+-----------+-----------------+-----------------+
| InventoryID | ProductID | QuantityInStock | LastStockUpdate |
+-------------+-----------+-----------------+-----------------+
|          11 |        11 |               8 | 2024-02-01      |
|          12 |        12 |              15 | 2024-02-02      |
|          13 |        13 |              10 | 2024-02-03      |
|          14 |        14 |               5 | 2024-02-04      |
|          15 |        15 |              12 | 2024-02-05      |
|          16 |        16 |               7 | 2024-02-06      |
|          17 |        17 |              10 | 2024-02-07      |
|          18 |        18 |               4 | 2024-02-08      |
|          19 |        19 |              18 | 2024-02-09      |
|          20 |        20 |               6 | 2024-02-10      |
+-------------+-----------+-----------------+-----------------+
10 rows in set (0.00 sec)
```

(Inventory Table)

Tasks 2: Select, Where, Between, AND, LIKE

1. Write an SQL query to retrieve the names and emails of all customers.

```
mysql> select concat(Firstname,' ',LastName) as Name ,Email from Customers;
+--------------+-------------------------+
| Name         | Email                   |
+--------------+-------------------------+
| Aarav Patel  | aarav.patel@example.com  |
| Aditi Sharma | aditi.sharma@example.com |
| Arjun Verma  | arjun.verma@example.com  |
| Avni Gupta   | avni.gupta@example.com   |
| Rahul Singh  | rahul.singh@example.com  |
| Neha Yadav   | neha.yadav@example.com   |
| Vedant Kumar | vedant.kumar@example.com |
| Isha Shah    | isha.shah@example.com    |
| Kabir Joshi  | kabir.joshi@example.com  |
| Ananya Das   | ananya.das@example.com   |
+--------------+-------------------------+
10 rows in set (0.01 sec)
```

2. Write an SQL query to list all orders with their order dates and corresponding customer names.

```
mysql> SELECT
    ->     OrderDate,
    ->     CONCAT(FirstName, ' ', LastName) AS Name
    -> FROM
    ->     Orders
    -> JOIN
    ->     Customers ON Orders.CustomerID = Customers.CustomerID;
+------------+--------------+
| OrderDate  | Name         |
+------------+--------------+
| 2024-02-01 | Aarav Patel  |
| 2024-02-02 | Aditi Sharma |
| 2024-02-03 | Arjun Verma  |
| 2024-02-04 | Avni Gupta   |
| 2024-02-05 | Rahul Singh  |
| 2024-02-06 | Neha Yadav   |
| 2024-02-07 | Vedant Kumar |
| 2024-02-08 | Isha Shah    |
| 2024-02-09 | Kabir Joshi  |
| 2024-02-10 | Ananya Das   |
+------------+--------------+
10 rows in set (0.00 sec)
```

3. Write an SQL query to insert a new customer record into the "Customers" table. Include customer information such as name, email, and address.

```
mysql> INSERT INTO Customers
    -> VALUES (21 , 'Vaybhav' , 'Sharma' , 'vaybhav.sharma@example.com' , '1234567890' , ' 123 Nari colony ');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Customers;
+------------+-----------+----------+---------------------------+------------+------------------+
| CustomerID | FirstName | LastName | Email                     | Phone      | Address          |
+------------+-----------+----------+---------------------------+------------+------------------+
|         11 | Aarav     | Patel    | aarav.patel@example.com   | 9876543210 | 456 Tulsi Lane   |
|         12 | Aditi     | Sharma   | aditi.sharma@example.com  | 7778889999 | 789 Ganges Street |
|         13 | Arjun     | Verma    | arjun.verma@example.com   | 5554443333 | 234 Himalaya Road |
|         14 | Avni      | Gupta    | avni.gupta@example.com    | 2221110000 | 567 Taj Avenue   |
|         15 | Rahul     | Singh    | rahul.singh@example.com   | 6667778888 | 890 Mango Lane   |
|         16 | Neha      | Yadav    | neha.yadav@example.com    | 1112223333 | 123 Rose Garden  |
|         17 | Vedant    | Kumar    | vedant.kumar@example.com  | 9998887777 | 678 Lotus Street |
|         18 | Isha      | Shah     | isha.shah@example.com     | 3332221111 | 345 Peacock Lane |
|         19 | Kabir     | Joshi    | kabir.joshi@example.com   | 8889990000 | 456 Mango Avenue |
|         20 | Ananya    | Das      | ananya.das@example.com    | 4445556666 | 789 Jasmine Road |
|         21 | Vaybhav   | Sharma   | vaybhav.sharma@example.com | 1234567890 |  123 Nari colony |
+------------+-----------+----------+---------------------------+------------+------------------+
11 rows in set (0.00 sec)
```

4. Write an SQL query to update the prices of all electronic gadgets in the "Products" table by increasing them by 10%.

```
mysql> UPDATE Products
    -> SET price = price + price*0.1;
Query OK, 10 rows affected, 10 warnings (0.01 sec)
Rows matched: 10  Changed: 10  Warnings: 10

mysql> Select * from Products;
+-----------+---------------------------+--------------------+--------+
| ProductID | ProductName               | Description        | Price  |
+-----------+---------------------------+--------------------+--------+
|        11 | Smartphone                | Electronic Gadgets | 879.99 |
|        12 | Air Conditioner           | Home Appliances    | 549.99 |
|        13 | Smart Refrigerator        | Electronic Gadgets | 989.99 |
|        14 | Smart Washing Machine     | Electronic Gadgets |  43.99 |
|        15 | Convection Microwave Oven | Kitchen Appliances | 141.56 |
|        16 | Laptop                    | Electronic Gadgets |  65.33 |
|        17 | Bluetooth Speaker         | Electronic Gadgets |  87.11 |
|        18 | Robotic Vacuum Cleaner    | Electronic Gadgets | 163.34 |
|        19 | Steam Iron                | Home Appliances    |  32.66 |
|        20 | Professional Hair Dryer   | Beauty Appliances  |  43.55 |
+-----------+---------------------------+--------------------+--------+
10 rows in set (0.00 sec)
```

5. Write an SQL query to delete a specific order and its associated order details from the "Orders" and "OrderDetails" tables. Allow users to input the order ID as a parameter.

```
mysql> SET @OrderIDToDelete = 118;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM OrderDetails WHERE OrderID = @OrderIDToDelete;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM OrderDetails;
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|            11 |     111 |        11 |        1 |
|            12 |     111 |        12 |        2 |
|            13 |     112 |        13 |        1 |
|            14 |     113 |        14 |        3 |
|            15 |     113 |        15 |        1 |
|            16 |     114 |        16 |        2 |
|            17 |     115 |        17 |        1 |
|            18 |     116 |        18 |        2 |
|            19 |     117 |        19 |        1 |
+---------------+---------+-----------+----------+
9 rows in set (0.00 sec)

mysql> DELETE FROM Orders WHERE OrderID = @OrderIDToDelete;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Orders;
+---------+------------+------------+-------------+
| OrderID | CustomerID | OrderDate  | TotalAmount |
+---------+------------+------------+-------------+
|     111 |         11 | 2024-02-01 |     1200.00 |
|     112 |         12 | 2024-02-02 |      850.00 |
|     113 |         13 | 2024-02-03 |      650.00 |
|     114 |         14 | 2024-02-04 |     1100.00 |
|     115 |         15 | 2024-02-05 |      450.00 |
|     116 |         16 | 2024-02-06 |     1050.00 |
|     117 |         17 | 2024-02-07 |      400.00 |
|     119 |         19 | 2024-02-09 |      600.00 |
|     120 |         20 | 2024-02-10 |      950.00 |
+---------+------------+------------+-------------+
9 rows in set (0.00 sec)
```

6. Write an SQL query to insert a new order into the "Orders" table. Include the customer ID, order date, and any other necessary information.

```
mysql> INSERT INTO Orders
    -> VALUES (121, 14, '2024-02-15', 500.00);
Query OK, 1 row affected (0.01 sec)

mysql> Select * from Orders;
+---------+------------+------------+-------------+
| OrderID | CustomerID | OrderDate  | TotalAmount |
+---------+------------+------------+-------------+
|     111 |         11 | 2024-02-01 |     1200.00 |
|     112 |         12 | 2024-02-02 |      850.00 |
|     113 |         13 | 2024-02-03 |      650.00 |
|     114 |         14 | 2024-02-04 |     1100.00 |
|     115 |         15 | 2024-02-05 |      450.00 |
|     116 |         16 | 2024-02-06 |     1050.00 |
|     117 |         17 | 2024-02-07 |      400.00 |
|     119 |         19 | 2024-02-09 |      600.00 |
|     120 |         20 | 2024-02-10 |      950.00 |
|     121 |         14 | 2024-02-15 |      500.00 |
+---------+------------+------------+-------------+
10 rows in set (0.00 sec)
```

7. Write an SQL query to update the contact information (e.g., email and address) of a specific customer in the "Customers" table. Allow users to input the customer ID and new contact information.

```
mysql> SET @CustomerIDToUpdate = 14;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @NewEmail = 'new.email@example.com';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @NewAddress = '123 New Street';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @NewPhone = '1112220000';
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE Customers
    -> SET Email = @NewEmail,
    -> Phone = @NewPhone, Address = @NewAddress
    -> WHERE CustomerID = @CustomerIDToUpdate;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Customers;
+------------+-----------+----------+--------------------------+------------+------------------+
| CustomerID | FirstName | LastName | Email                    | Phone      | Address          |
+------------+-----------+----------+--------------------------+------------+------------------+
|         11 | Aarav     | Patel    | aarav.patel@example.com  | 9876543210 | 456 Tulsi Lane   |
|         12 | Aditi     | Sharma   | aditi.sharma@example.com | 7778889999 | 789 Ganges Street |
|         13 | Arjun     | Verma    | arjun.verma@example.com  | 5554443333 | 234 Himalaya Road |
|         14 | Avni      | Gupta    | new.email@example.com    | 1112220000 | 123 New Street   |
|         15 | Rahul     | Singh    | rahul.singh@example.com  | 6667778888 | 890 Mango Lane   |
|         16 | Neha      | Yadav    | neha.yadav@example.com   | 1112223333 | 123 Rose Garden  |
|         17 | Vedant    | Kumar    | vedant.kumar@example.com | 9998887777 | 678 Lotus Street |
|         18 | Isha      | Shah     | isha.shah@example.com    | 3332221111 | 345 Peacock Lane |
|         19 | Kabir     | Joshi    | kabir.joshi@example.com  | 8889990000 | 456 Mango Avenue |
|         20 | Ananya    | Das      | ananya.das@example.com   | 4445556666 | 789 Jasmine Road |
|         21 | Vaybhav   | Sharma   | vaybhav.sharma@example.com | 1234567890 | 123 Nari Colony |
+------------+-----------+----------+--------------------------+------------+------------------+
11 rows in set (0.00 sec)
```

8. Write an SQL query to recalculate and update the total cost of each order in the "Orders" table based on the prices and quantities in the "OrderDetails" table.

```
mysql> UPDATE Orders
    -> SET TotalAmount = (
    ->     SELECT SUM(Products.Price * OrderDetails.Quantity)
    ->     FROM OrderDetails
    ->     JOIN Products ON OrderDetails.ProductID = Products.ProductID
    ->     WHERE OrderDetails.OrderID = Orders.OrderID
    -> )
    -> WHERE OrderID IN (SELECT DISTINCT OrderID FROM OrderDetails);
Query OK, 7 rows affected (0.01 sec)
Rows matched: 7  Changed: 7  Warnings: 0

mysql> SELECT * FROM Orders;
+---------+------------+------------+-------------+
| OrderID | CustomerID | OrderDate  | TotalAmount |
+---------+------------+------------+-------------+
|     111 |         11 | 2024-02-01 |     1979.97 |
|     112 |         12 | 2024-02-02 |      989.99 |
|     113 |         13 | 2024-02-03 |     2287.96 |
|     114 |         14 | 2024-02-04 |      131.98 |
|     115 |         15 | 2024-02-05 |       87.99 |
|     116 |         16 | 2024-02-06 |      329.98 |
|     117 |         17 | 2024-02-07 |       32.99 |
|     119 |         19 | 2024-02-09 |      600.00 |
|     120 |         20 | 2024-02-10 |      950.00 |
|     121 |         14 | 2024-02-15 |      500.00 |
+---------+------------+------------+-------------+
10 rows in set (0.00 sec)
```

9. Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID as a parameter.

```
mysql> SET @CustomerIDToDelete = 14;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM OrderDetails
    -> WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = @CustomerIDToDelete);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM OrderDetails;
+---------------+---------+-----------+----------+
| OrderDetailID | OrderID | ProductID | Quantity |
+---------------+---------+-----------+----------+
|            11 |     111 |        11 |        1 |
|            12 |     111 |        12 |        2 |
|            13 |     112 |        13 |        1 |
|            14 |     113 |        14 |        3 |
|            15 |     113 |        15 |        1 |
|            17 |     115 |        17 |        1 |
|            18 |     116 |        18 |        2 |
|            19 |     117 |        19 |        1 |
+---------------+---------+-----------+----------+
8 rows in set (0.00 sec)
```

(OrderDetails)

```
mysql> DELETE FROM Orders
    -> WHERE CustomerID = @CustomerIDToDelete;
Query OK, 2 rows affected (0.01 sec)

mysql> SELECT * FROM ORDERS;
+---------+------------+------------+-------------+
| OrderID | CustomerID | OrderDate  | TotalAmount |
+---------+------------+------------+-------------+
|     111 |         11 | 2024-02-01 |     1979.97 |
|     112 |         12 | 2024-02-02 |      989.99 |
|     113 |         13 | 2024-02-03 |     2287.96 |
|     115 |         15 | 2024-02-05 |       87.99 |
|     116 |         16 | 2024-02-06 |      329.98 |
|     117 |         17 | 2024-02-07 |       32.99 |
|     119 |         19 | 2024-02-09 |      600.00 |
|     120 |         20 | 2024-02-10 |      950.00 |
+---------+------------+------------+-------------+
8 rows in set (0.00 sec)
```

(OrderDetails)

10. Write an SQL query to insert a new electronic gadget product into the "Products" table, including product name, category, price, and any other relevant details.

```
mysql> INSERT INTO Products
    -> VALUES (21, 'Smartwatch', 'Electronic Gadgets', 149.99);
Query OK, 1 row affected (0.01 sec)

mysql> Select * from Products;
+-----------+--------------------------+--------------------+--------+
| ProductID | ProductName              | Description        | Price  |
+-----------+--------------------------+--------------------+--------+
|        11 | Smartphone               | Electronic Gadgets | 879.99 |
|        12 | Air Conditioner          | Home Appliances    | 549.99 |
|        13 | Smart Refrigerator       | Electronic Gadgets | 989.99 |
|        14 | Smart Washing Machine    | Electronic Gadgets |  43.99 |
|        15 | Convection Microwave Oven | Kitchen Appliances | 141.56 |
|        16 | Laptop                   | Electronic Gadgets |  65.33 |
|        17 | Bluetooth Speaker        | Electronic Gadgets |  87.11 |
|        18 | Robotic Vacuum Cleaner   | Electronic Gadgets | 163.34 |
|        19 | Steam Iron               | Home Appliances    |  32.66 |
|        20 | Professional Hair Dryer  | Beauty Appliances  |  43.55 |
|        21 | Smartwatch               | Electronic Gadgets | 149.99 |
+-----------+--------------------------+--------------------+--------+
11 rows in set (0.00 sec)
```

11. Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from "Pending" to "Shipped"). Allow users to input the order ID and the new status.

```
mysql> SELECT
    ->     OrderID,
    ->     OrderDate,
    ->     CASE
    ->         WHEN DATEDIFF(CURDATE(), OrderDate) >= 2 THEN 'Shipped'
    ->         ELSE 'Pending'
    ->     END AS OrderStatus
    -> FROM
    ->     Orders;
+---------+------------+-------------+
| OrderID | OrderDate  | OrderStatus |
+---------+------------+-------------+
|     111 | 2024-02-01 | Pending     |
|     112 | 2024-02-02 | Pending     |
|     113 | 2024-02-03 | Pending     |
|     115 | 2024-02-05 | Pending     |
|     116 | 2024-02-06 | Pending     |
|     117 | 2024-02-07 | Pending     |
|     119 | 2024-02-09 | Pending     |
|     120 | 2024-02-10 | Pending     |
+---------+------------+-------------+
8 rows in set (0.01 sec)
```

12. Write an SQL query to calculate and update the number of orders placed by each customer in the "Customers" table based on the data in the "Orders" table.

```
mysql> ALTER TABLE Customers
    -> ADD COLUMN NumberOfOrders INT;
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> UPDATE Customers
    -> SET NumberOfOrders = (
    ->     SELECT COUNT(OrderID)
    ->     FROM Orders
    ->     WHERE Customers.CustomerID = Orders.CustomerID
    -> );
Query OK, 11 rows affected (0.02 sec)
Rows matched: 11  Changed: 11  Warnings: 0

mysql> SELECT * FROM Customers;
+------------+-----------+----------+--------------------------+------------+------------------+----------------+
| CustomerID | FirstName | LastName | Email                    | Phone      | Address          | NumberOfOrders |
+------------+-----------+----------+--------------------------+------------+------------------+----------------+
|         11 | Aarav     | Patel    | aarav.patel@example.com  | 9876543210 | 456 Tulsi Lane   |              1 |
|         12 | Aditi     | Sharma   | aditi.sharma@example.com | 7778889999 | 789 Ganges Street |              1 |
|         13 | Arjun     | Verma    | arjun.verma@example.com  | 5554443333 | 234 Himalaya Road |              1 |
|         14 | Avni      | Gupta    | new.email@example.com    | 1112220000 | 123 New Street   |              0 |
|         15 | Rahul     | Singh    | rahul.singh@example.com  | 6667778888 | 890 Mango Lane   |              1 |
|         16 | Neha      | Yadav    | neha.yadav@example.com   | 1112223333 | 123 Rose Garden  |              1 |
|         17 | Vedant    | Kumar    | vedant.kumar@example.com | 9998887777 | 678 Lotus Street |              1 |
|         18 | Isha      | Shah     | isha.shah@example.com    | 3332221111 | 345 Peacock Lane |              0 |
|         19 | Kabir     | Joshi    | kabir.joshi@example.com  | 8889990000 | 456 Mango Avenue |              1 |
|         20 | Ananya    | Das      | ananya.das@example.com   | 4445556666 | 789 Jasmine Road |              1 |
|         21 | Vaybhav   | Sharma   | vaybhav.sharma@example.com | 1234567890 | 123 Nari Colony  |              0 |
+------------+-----------+----------+--------------------------+------------+------------------+----------------+
11 rows in set (0.00 sec)
```

Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write an SQL query to retrieve a list of all orders along with customer information (e.g., customer name) for each order.

```
mysql> SELECT
    -> Orders.OrderID,
    -> CONCAT(Customers.FirstName,' ',Customers.LastName) as CustomerName,
    -> Customers.Phone,
    -> Orders.OrderDate,
    -> Orders.TotalAmount
    -> From Orders Join Customers on Orders.CustomerID = Customers.CustomerID;
+---------+--------------+------------+------------+-------------+
| OrderID | CustomerName | Phone      | OrderDate  | TotalAmount |
+---------+--------------+------------+------------+-------------+
|     111 | Aarav Patel  | 9876543210 | 2024-02-01 |     1979.97 |
|     112 | Aditi Sharma | 7778889999 | 2024-02-02 |      989.99 |
|     113 | Arjun Verma  | 5554443333 | 2024-02-03 |     2287.96 |
|     115 | Rahul Singh  | 6667778888 | 2024-02-05 |       87.99 |
|     116 | Neha Yadav   | 1112223333 | 2024-02-06 |      329.98 |
|     117 | Vedant Kumar | 9998887777 | 2024-02-07 |       32.99 |
|     119 | Kabir Joshi  | 8889990000 | 2024-02-09 |      600.00 |
|     120 | Ananya Das   | 4445556666 | 2024-02-10 |      950.00 |
+---------+--------------+------------+------------+-------------+
8 rows in set (0.00 sec)
```

2. Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.

```
mysql> SELECT
    ->     Products.ProductName,
    ->     SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
    -> FROM
    ->     OrderDetails
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID
    -> WHERE
    ->     Products.Description = 'Electronic Gadgets'
    -> GROUP BY
    ->     Products.ProductName;
+-----------------------+--------------+
| ProductName           | TotalRevenue |
+-----------------------+--------------+
| Smartphone            |       879.99 |
| Smart Refrigerator    |       989.99 |
| Smart Washing Machine |       131.97 |
| Bluetooth Speaker     |        87.11 |
| Robotic Vacuum Cleaner |      326.68 |
+-----------------------+--------------+
5 rows in set (0.01 sec)
```

3. Write an SQL query to list all customers who have made at least one purchase. Include their names and contact information.

```
mysql> SELECT
    ->     Customers.FirstName,
    ->     Customers.LastName,
    ->     Customers.Email,
    ->     Customers.Phone,
    ->     Customers.Address
    -> FROM
    ->     Customers
    -> WHERE
    ->     Customers.CustomerID IN (
    ->         SELECT DISTINCT CustomerID
    ->         FROM Orders);
+-----------+----------+-------------------------+------------+------------------+
| FirstName | LastName | Email                   | Phone      | Address          |
+-----------+----------+-------------------------+------------+------------------+
| Aarav     | Patel    | aarav.patel@example.com | 9876543210 | 456 Tulsi Lane   |
| Aditi     | Sharma   | aditi.sharma@example.com | 7778889999 | 789 Ganges Street |
| Arjun     | Verma    | arjun.verma@example.com | 5554443333 | 234 Himalaya Road |
| Rahul     | Singh    | rahul.singh@example.com | 6667778888 | 890 Mango Lane   |
| Neha      | Yadav    | neha.yadav@example.com  | 1112223333 | 123 Rose Garden  |
| Vedant    | Kumar    | vedant.kumar@example.com | 9998887777 | 678 Lotus Street |
| Kabir     | Joshi    | kabir.joshi@example.com | 8889990000 | 456 Mango Avenue |
| Ananya    | Das      | ananya.das@example.com  | 4445556666 | 789 Jasmine Road |
+-----------+----------+-------------------------+------------+------------------+
8 rows in set (0.01 sec)
```

4. Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.

```
mysql> SELECT
    ->     Products.ProductName,
    ->     SUM(OrderDetails.Quantity) AS TotalQuantityOrdered
    -> FROM
    ->     OrderDetails
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID
    -> WHERE
    ->     Products.Description = 'Electronic Gadgets'
    -> GROUP BY
    ->     Products.ProductName
    -> ORDER BY
    ->     TotalQuantityOrdered DESC
    -> LIMIT 1;
+----------------------+----------------------+
| ProductName          | TotalQuantityOrdered |
+----------------------+----------------------+
| Smart Washing Machine |                    3 |
+----------------------+----------------------+
1 row in set (0.00 sec)
```

5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding categories.

```
mysql> SELECT
    ->     ProductName,
    ->     Price,
    ->     Description as Category
    -> FROM
    ->     Products
    -> WHERE
    ->     Description = 'Electronic Gadgets';
+-----------------------+--------+--------------------+
| ProductName           | Price  | Category           |
+-----------------------+--------+--------------------+
| Smartphone            | 879.99 | Electronic Gadgets |
| Smart Refrigerator    | 989.99 | Electronic Gadgets |
| Smart Washing Machine |  43.99 | Electronic Gadgets |
| Laptop                |  65.33 | Electronic Gadgets |
| Bluetooth Speaker     |  87.11 | Electronic Gadgets |
| Robotic Vacuum Cleaner| 163.34 | Electronic Gadgets |
| Smartwatch            | 149.99 | Electronic Gadgets |
+-----------------------+--------+--------------------+
7 rows in set (0.01 sec)
```

6. Write an SQL query to calculate the average order value for each customer. Include the customer's name and their average order value.

```
mysql> SELECT
    ->     CONCAT(Customers.FirstName,' ',Customers.LastName) as CustomerName,
    ->     AVG(Orders.TotalAmount) AS AverageOrderValue
    -> FROM
    ->     Customers
    -> JOIN
    ->     Orders ON Customers.CustomerID = Orders.CustomerID
    -> GROUP BY
    ->     Customers.CustomerID, Customers.FirstName, Customers.LastName;
+--------------+-------------------+
| CustomerName | AverageOrderValue |
+--------------+-------------------+
| Aarav Patel  |       1979.970000 |
| Aditi Sharma |        989.990000 |
| Arjun Verma  |       2287.960000 |
| Rahul Singh  |         87.990000 |
| Neha Yadav   |        329.980000 |
| Vedant Kumar |         32.990000 |
| Kabir Joshi  |        600.000000 |
| Ananya Das   |        950.000000 |
+--------------+-------------------+
8 rows in set (0.00 sec)
```

7. Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.

```
mysql> SELECT
    ->     Orders.OrderID,
    ->     Customers.FirstName,
    ->     Customers.LastName,
    ->     Customers.Email,
    ->     Customers.Phone,
    ->     Customers.Address,
    ->     SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
    -> FROM
    ->     Orders
    -> JOIN
    ->     Customers ON Orders.CustomerID = Customers.CustomerID
    -> JOIN
    ->     OrderDetails ON Orders.OrderID = OrderDetails.OrderID
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID
    -> GROUP BY
    ->     Orders.OrderID, Customers.FirstName, Customers.LastName, Customers.Email, Customers.Phone, Customers.Address
    -> ORDER BY
    ->     TotalRevenue DESC
    -> LIMIT 1;
+---------+-----------+----------+-----------------------+------------+---------------+--------------+
| OrderID | FirstName | LastName | Email                 | Phone      | Address       | TotalRevenue |
+---------+-----------+----------+-----------------------+------------+---------------+--------------+
|     111 | Aarav     | Patel    | aarav.patel@example.com | 9876543210 | 456 Tulsi Lane |      1979.97 |
+---------+-----------+----------+-----------------------+------------+---------------+--------------+
1 row in set (0.00 sec)
```

8. Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.

```
mysql> SELECT
    ->     Products.ProductName,
    ->     COUNT(OrderDetails.OrderDetailID) AS NumberOfOrders
    -> FROM
    ->     Products
    -> LEFT JOIN
    ->     OrderDetails ON Products.ProductID = OrderDetails.ProductID
    -> WHERE
    ->     Products.Description = 'Electronic Gadgets'
    -> GROUP BY
    ->     Products.ProductName;
+-----------------------+----------------+
| ProductName           | NumberOfOrders |
+-----------------------+----------------+
| Smartphone            |              1 |
| Smart Refrigerator    |              1 |
| Smart Washing Machine |              1 |
| Laptop                |              0 |
| Bluetooth Speaker     |              1 |
| Robotic Vacuum Cleaner |              1 |
| Smartwatch            |              0 |
+-----------------------+----------------+
7 rows in set (0.01 sec)
```

9. Write an SQL query to find the order with the highest total revenue. Include the order ID, customer information, and the total revenue.

```
mysql> SET @EnterProductName = 'Smartphone';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT
    ->     Customers.CustomerID,
    ->     Customers.FirstName,
    ->     Customers.LastName,
    ->     Customers.Email,
    ->     Customers.Phone,
    ->     Customers.Address
    -> FROM
    ->     Customers
    -> JOIN
    ->     Orders ON Customers.CustomerID = Orders.CustomerID
    -> JOIN
    ->     OrderDetails ON Orders.OrderID = OrderDetails.OrderID
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID
    -> WHERE
    ->     Products.ProductName = @EnterProductName;
+------------+-----------+----------+------------------------+------------+----------------+
| CustomerID | FirstName | LastName | Email                  | Phone      | Address        |
+------------+-----------+----------+------------------------+------------+----------------+
|         11 | Aarav     | Patel    | aarav.patel@example.com | 9876543210 | 456 Tulsi Lane |
+------------+-----------+----------+------------------------+------------+----------------+
1 row in set (0.00 sec)
```

10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.

```
mysql> SET @EnterStartDate = '2024-02-03';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @EnterEndDate = '2024-02-08';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT
    ->     SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
    -> FROM
    ->     Orders
    -> JOIN
    ->     OrderDetails ON Orders.OrderID = OrderDetails.OrderID
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID
    -> WHERE
    ->     Orders.OrderDate BETWEEN @EnterStartDate AND @EnterEndDate;
+--------------+
| TotalRevenue |
+--------------+
|       719.98 |
+--------------+
1 row in set (0.01 sec)
```

Task 4. Subquery and its type:

1. Write an SQL query to find out which customers have not placed any orders.

```
mysql> SELECT * FROM Customers
    -> LEFT JOIN
    -> Orders ON Customers.CustomerID = Orders.CustomerID
    -> WHERE
    -> Orders.OrderID IS NULL;
+------------+-----------+----------+-------------------------+------------+-----------------+---------------+
-------+
| CustomerID | FirstName | LastName | Email                   | Phone      | Address         | NumberOfOrders |
Amount |
+------------+-----------+----------+-------------------------+------------+-----------------+---------------+
-------+
|         14 | Avni      | Gupta    | new.email@example.com   | 1112220000 | 123 New Street  |              0 |
  NULL |
|         18 | Isha      | Shah     | isha.shah@example.com   | 3332221111 | 345 Peacock Lane |             0 |
  NULL |
|         21 | Vaybhav   | Sharma   | vaybhav.sharma@example.com | 1234567890 | 123 Nari Colony |            0 |
  NULL |
+------------+-----------+----------+-------------------------+------------+-----------------+---------------+
-------+
3 rows in set (0.01 sec)
```

2. Write an SQL query to find the total number of products available for sale.

```
mysql> SELECT SUM(QuantityInStock) as TotalProductAvailable
    -> FROM Inventory;
+-----------------------+
| TotalProductAvailable |
+-----------------------+
|                    95 |
+-----------------------+
1 row in set (0.00 sec)
```

3. Write an SQL query to calculate the total revenue generated by TechShop.

```
mysql> SELECT
    ->     SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
    -> FROM
    ->     OrderDetails
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID;
+--------------+
| TotalRevenue |
+--------------+
|      3689.94 |
+--------------+
1 row in set (0.01 sec)
```

4. Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter.

```
mysql> SELECT
    ->     Products.Description,
    ->     AVG(OrderDetails.Quantity)
    -> FROM
    ->     OrderDetails
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID
    -> GROUP BY
    ->     Products.Description;
+--------------------+----------------------------+
| Description        | AVG(OrderDetails.Quantity) |
+--------------------+----------------------------+
| Electronic Gadgets |                     1.6000 |
| Home Appliances    |                     1.5000 |
| Kitchen Appliances |                     1.0000 |
+--------------------+----------------------------+
3 rows in set (0.01 sec)
```

5. Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.

```
mysql> SET @EnterCustomeID = 13;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT
    ->     CONCAT(Customers.FirstName, ' ', Customers.LastName) AS CustomerName,
    ->     SUM(Customers.NumberOfOrders * Products.Price) AS TotalRevenue
    -> FROM
    ->     Customers
    -> JOIN
    ->     Orders ON Customers.CustomerID = Orders.CustomerID
    -> JOIN
    ->     OrderDetails ON Orders.OrderID = OrderDetails.OrderID
    -> JOIN
    ->     Products ON OrderDetails.ProductID = Products.ProductID
    -> WHERE
    ->     Customers.CustomerID = @EnterCustomerID
    -> GROUP BY
    ->     Customers.CustomerID;
+--------------+--------------+
| CustomerName | TotalRevenue |
+--------------+--------------+
| Arjun Verma  |       185.55 |
+--------------+--------------+
1 row in set (0.01 sec)
```

6. Write an SQL query to find the customers who have placed the most orders. List their names and the number of orders they've placed.

```
mysql> SELECT
    ->     CONCAT(FirstName, ' ', LastName) AS CustomerName,
    ->     COUNT(OrderID) AS NumberOfOrders
    -> FROM
    ->     Customers
    -> JOIN
    ->     Orders ON Customers.CustomerID = Orders.CustomerID
    -> GROUP BY
    ->     Customers.CustomerID
    -> HAVING
    ->     NumberOfOrders = (
    ->         SELECT
    ->             MAX(OrderCount)
    ->         FROM (
    ->             SELECT
    ->                 COUNT(OrderID) AS OrderCount
    ->             FROM
    ->                 Customers
    ->             JOIN
    ->                 Orders ON Customers.CustomerID = Orders.CustomerID
    ->             GROUP BY
    ->                 Customers.CustomerID
    ->         ) AS OrderCounts);
+--------------+-----------------+
| CustomerName | NumberOfOrders |
+--------------+-----------------+
| Aarav Patel  |               1 |
| Aditi Sharma |               1 |
| Arjun Verma  |               1 |
| Rahul Singh  |               1 |
| Neha Yadav   |               1 |
| Vedant Kumar |               1 |
| Kabir Joshi  |               1 |
| Ananya Das   |               1 |
+--------------+-----------------+
8 rows in set (0.01 sec)
```

7. Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.

```
mysql> SELECT
    ->  Products.Description as CategoryOfProduct,
    ->  SUM(OrderDetails.Quantity) as TotalQuantity
    -> From OrderDetails
    -> JOIN Products on OrderDetails.ProductID = Products.ProductID
    -> GROUP BY Products.Description
    -> ORDER BY TotalQuantity DESC
    -> LIMIT 1;
+-------------------+---------------+
| CategoryOfProduct  | TotalQuantity |
+-------------------+---------------+
| Electronic Gadgets |             8 |
+-------------------+---------------+
1 row in set (0.00 sec)
```

8. Write an SQL query to find the customer who has spent the most money (highest total revenue) on electronic gadgets. List their name and total spending.

```
mysql> SELECT
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     SUM(P.Price * OD.Quantity) AS TotalSpending
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Orders O ON C.CustomerID = O.CustomerID
    -> JOIN
    ->     OrderDetails OD ON O.OrderID = OD.OrderID
    -> JOIN
    ->     Products P ON OD.ProductID = P.ProductID
    -> WHERE
    ->     P.Description = 'Electronic Gadgets'
    -> GROUP BY
    ->     C.CustomerID
    -> ORDER BY
    ->     TotalSpending DESC
    -> LIMIT 1;
+--------------+--------------+
| CustomerName | TotalSpending |
+--------------+--------------+
| Aditi Sharma |        989.99 |
+--------------+--------------+
1 row in set (0.00 sec)
```

9. Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.

```
mysql> SELECT
    ->     C.CustomerID,
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     COUNT(O.OrderID) AS NumberOfOrders,
    ->     SUM(P.Price * OD.Quantity) AS TotalRevenue,
    ->     AVG(P.Price * OD.Quantity) AS AverageOrderValue
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Orders O ON C.CustomerID = O.CustomerID
    -> JOIN
    ->     OrderDetails OD ON O.OrderID = OD.OrderID
    -> JOIN
    ->     Products P ON OD.ProductID = P.ProductID
    -> GROUP BY
    ->     C.CustomerID
    -> ORDER BY
    ->     AverageOrderValue DESC;
+------------+--------------+----------------+--------------+-------------------+
| CustomerID | CustomerName | NumberOfOrders | TotalRevenue | AverageOrderValue |
+------------+--------------+----------------+--------------+-------------------+
|         12 | Aditi Sharma |              1 |       989.99 |        989.990000 |
|         11 | Aarav Patel  |              2 |      1979.97 |        989.985000 |
|         16 | Neha Yadav   |              1 |       326.68 |        326.680000 |
|         13 | Arjun Verma  |              2 |       273.53 |        136.765000 |
|         15 | Rahul Singh  |              1 |        87.11 |         87.110000 |
|         17 | Vedant Kumar |              1 |        32.66 |         32.660000 |
+------------+--------------+----------------+--------------+-------------------+
6 rows in set (0.00 sec)
```

10. Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.

```
mysql> SELECT
    ->     C.CustomerID,
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     COUNT(O.OrderID) AS OrderCount
    -> FROM
    ->     Customers C
    -> LEFT JOIN
    ->     Orders O ON C.CustomerID = O.CustomerID
    -> GROUP BY
    ->     C.CustomerID;
+------------+----------------+------------+
| CustomerID | CustomerName   | OrderCount |
+------------+----------------+------------+
|         11 | Aarav Patel    |          1 |
|         12 | Aditi Sharma   |          1 |
|         13 | Arjun Verma    |          1 |
|         14 | Avni Gupta     |          0 |
|         15 | Rahul Singh    |          1 |
|         16 | Neha Yadav     |          1 |
|         17 | Vedant Kumar   |          1 |
|         18 | Isha Shah      |          0 |
|         19 | Kabir Joshi    |          1 |
|         20 | Ananya Das     |          1 |
|         21 | Vaybhav Sharma |          0 |
+------------+----------------+------------+
11 rows in set (0.01 sec)
```